

FACIALDATACOLLECTOR: A NOVEL HEURISTIC-BASED ALGORITHM FOR AUTOMATED VIDEO SEGMENTATION

1 INTRODUCTION

1.1 BACKGROUND OF RESEARCH AREA

Emotion recognition is the process of identifying emotions from facial expressions. Currently, empowering machines with emotion recognition abilities can be done via deep learning, which requires large amount of high quality data. The training process of developing an Artificial Intelligence (AI) with emotion recognition abilities can be represented as follows (Figure 1).

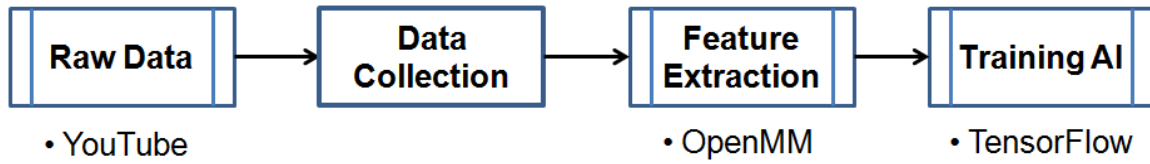


Figure 1: Training process of developing an AI with emotion-recognition abilities

In the stage of collecting raw data, video clips could be obtained from YouTube. There are also multiple tools which can facilitate in extracting features, such as OpenMM (Morales, 2017). To train an AI, Google's TensorFlow¹ can help researchers. However, in the data collection stage, there are no tools which are efficient in collecting high-quality data, which is important for visual feature extraction. While there are commercial projects which deal with facial tracking, such as Filmora Video Editor², there is no product which deals with the segmentation of videos by the presence of facial elements. This is crucial for movies, where there are many scene transitions.

1.2 PURPOSE OF RESEARCH AREA

The purpose of the research is to develop a fast and effective tool which enables the segmentation of videos based on whether there is a human face present in a frame. This provides visual features to determine emotions. The tool must be time-efficient to process videos of long durations quickly. It must also be effective to generate quality data for researchers: by reducing

¹ <https://www.tensorflow.org/>

² <https://filmora.wondershare.com/video-editor/>

the number of fragmented clips when processing frames of the same scene. The effectiveness of the program, FacialDataCollector, will also be compared with other methods of data collection.

2 PROGRAM DESIGN AND ALGORITHM

The program involves two main steps. The first step is to recognise the presence of a face in a frame. The second step is to determine whether the next frame continues from the previous

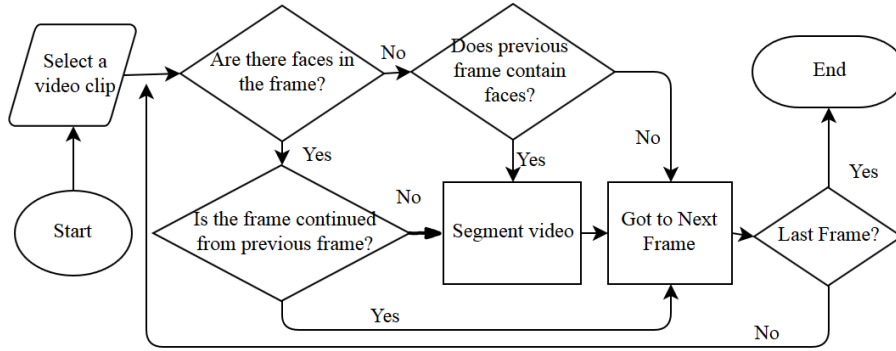


Figure 2: Workflow of FacialDataCollector program

frame. Two frames are *continuous* if they differ by a timing of 0.1 seconds, show similar scenes and contain the same set of faces. This concept of continuity can be

implemented by introducing an error function and an error threshold (ϵ), which will be discussed later. The logic of the program is presented in the workflow figure (Figure 2).

2.1 FACE DETECTION IN VIDEO

Haar feature-based cascade classifiers were used for detecting faces as they are effective tools for object detection. OpenCV (OpenCV, 2017) has four pre-trained classifiers which allow for the detection of frontal face elements. A frame will be said to contain a face if the program detects a face using any one of these four classifiers, which improves the accuracy of the program.

2.2 ERROR FUNCTION

To check whether two frames are continuous, an error function is implemented, which reports the error value between two adjacent frames. We will also set an error threshold (ϵ), which is the maximum error value we allow adjacent frames to keep. The derivation of the value of error threshold will be discussed in Section 3.1.

The error value between two adjacent frames depends on two main factors. It depends on the change in face positions and the change in the value of the pixels of the whole image.

2.2.1 ERROR FROM CHANGE OF FACE POSITION

To calculate the error from change in face positions, the sum of squares technique is used on the differences between the coordinates of the rectangles marking the faces. Let's say in frame f , a

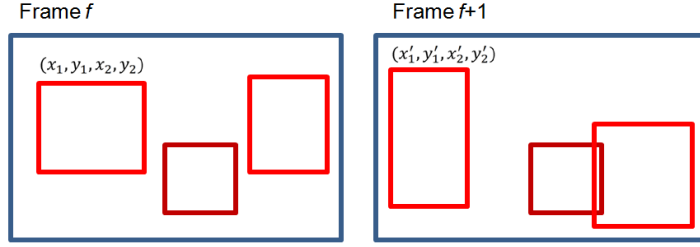


Figure 3: Red and brown rectangles are rectangles marking out faces.

If brown rectangles are in the same relative position,
then the face error between these two frames is 0.

face is covered by (x_1, y_1, x_2, y_2) , where x_1, y_1, x_2, y_2 denotes the bottom-left x-coordinate, bottom-left y-coordinate, upper-right x-coordinate and upper-right y-coordinate respectively. Suppose in the next frame f' , the coordinates are

transformed into (x'_1, y'_1, x'_2, y'_2) . Then the error is defined to be the sum of squares of the differences in the coordinates (Equation 1).

$$error_{face}(f) = \sum_{i=1,2} [(x'_i - x_i)^2 + (y'_i - y_i)^2]$$

Equation 1: Error from change in position of faces from two different frames

If there are multiple faces in f and multiple faces in f' (Figure 3), then the value of the error can be generalised to the minimum error between pairwise faces (Equation 2).

$$error_{face}(f) = \min_{g \in f, g' \in f'} \sum_{i=1,2} [(g'[x_i] - g[x_i])^2 + (g'[y_i] - g[y_i])^2]$$

Equation 2: Error from change in position of multiple faces from two different frames

2.2.2 ERROR FROM CHANGE OF BACKGROUND PIXEL VALUE

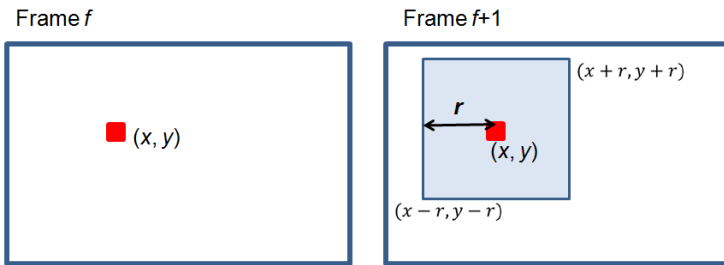


Figure 4: Graphical depiction of pixel error calculation method

For pixel error calculation, a pixel error function is proposed. Suppose we focus on a pixel at position (x, y) in a frame (Figure 4). In the next frame, the program searches in the range (x', y') where $x - r <$

$x' < x + r$ and $y - r < y' < y + r$ for the pixel with value closest to that of pixel at (x, y) . The pixel error will be the square of the difference in pixel value of the two pixels (Equation 3). The

value of r can be set arbitrarily to 5. The total pixel error of a frame can be found by summing over the error of all pixels in a frame (Equation 4).

$$error_{pixel}(x, y, f) = \min_{x-r < x' < x+r, y-r < y' < y+r} |pixelval(x', y', f+1) - pixelval(x, y, f)|^2$$

Equation 3: Pixel error as the minimum difference between two pixels in adjacent frames

$$error_{pixel}(f) = \sum error_{pixel}(x, y, f)$$

Equation 4: Total pixel error of a frame as the sum of all errors of pixels in a frame

2.2.3 COMBINED ERROR

$error_{face}$ calculates an error value from four coordinates while $error_{pixel}$ calculates an error value from across the whole frame. To make sure that error values are of similar order of magnitude, we introduce a function for calculating the combined error value (Equation 5).

$$error = error_{face} + \frac{error_{pixel}}{dimension\ of\ frame}$$

Equation 5: Combined error value as a function of face error and pixel error

3 PROGRAM EVALUATION AND VALIDATION

3.1 EPSILON DERIVATION

FacialDataCollector requires the setting of a error threshold (ϵ) to judge whether two frames are considered continuous. If epsilon is set as a number with high magnitude, then it is likely that non-continuous frames may be accidentally allowed and this will compromise the quality of data produced. However, if the value of epsilon is too low, then it is likely that continuous frames may be split up due to imperfections in the Haar cascade classifiers in recognising faces.

The videos used (Table 1) are obtained from YouTube. The effectiveness of each epsilon value is defined to be the number of clips produced using that epsilon value as the error threshold, divided by the expected number of clips produced. The expected number of clips to be produced is determined based on the judgement of a human observer. Therefore, the optimal value of epsilon should yield an effectiveness of close to 100% for every video.

Video No.	Video Name	Artiste	Duration/s	Start Timing	End Timing
1	Despacito	Luis Fonsi	30	00.32	01.02
2	I Don't Wanna Live Forever	ZAYN, Taylor Swift	30	00.08	00.38
3	Work from Home	Fifth Harmony	30	00.30	01.00
4	Chantaje	Shakira	30	00.00	00.30
5	Love Yourself	Justin Bieber	30	01.44	02.14

Table 1: Videos used in deriving the value of epsilon

Firstly, a sample order of magnitude test is carried out. Using the error function on sample frames of Video 1 (Despacito), there is a difference of two orders of magnitude for error values between continuous frames and between unrelated frames (Table 2). This shows that the error functions are good indicators for determining continuity between frames.

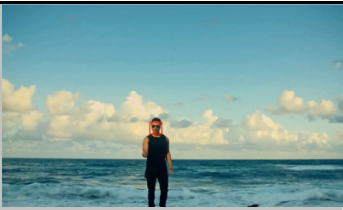
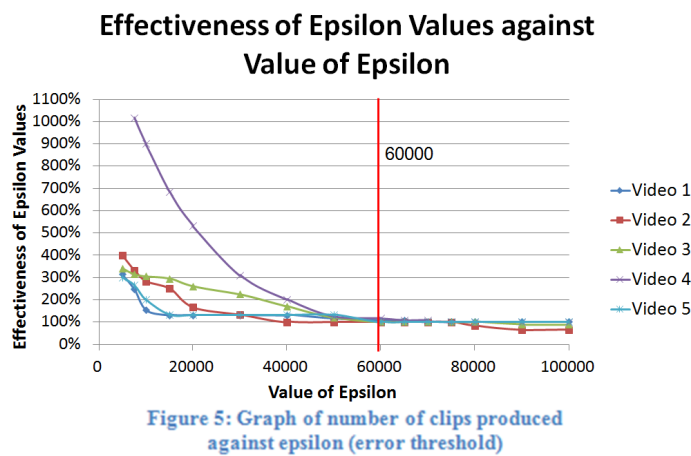
Time (s)	3.0	3.1 (continuous from 3.0)	10.2 (unrelated from 3.0)
Frames			
<i>error_{face}</i>	0	49	50690
<i>error_{pixel}</i> <i>dimension</i>	0	6940	57383
<i>error</i>	0	6989	108073

Table 2: Error between pair of continuous frames (3.0 and 3.1) and unrelated frames (3.0 and 10.2)



To obtain the optimal value of epsilon, the effectiveness of epsilon values is plotted against the value of epsilon. Results showed that as $\epsilon \rightarrow 60000$, the effectiveness converges to 100% (Figure 5), meaning frames which are supposed to be continuous are segmented together.

Also, the effectiveness of epsilon values are close to 100% in the range $\epsilon \in [60000, 75000]$ (Figure 5). This further shows that the error functions proposed are good indicators of continuity of frames because the error between unrelated frames is so large that the number of clips produced is unaffected for epsilon value in a large range. Epsilon is set as 60000 to ensure high-quality data is obtained, such that the number of clips obtained is close to expectation.

3.2 EVALUATION METHODOLOGY

We used the following movie clips from YouTube (Table 3). These clips have a duration of more than 90 minutes, which can provide an accurate judge of effectiveness.

S/N	Movie Title	Year	Director	Duration
1	Ilo Ilo	2013	Anthony Chen	99 minutes
2	Singapore Dreaming	2006	Colin Goh, Yen Yen Woo	105 minutes
3	Chicken Rice War	2000	David Leong, Suat Yen Lim	100 minutes

Table 3: Movies used for comparing methods of data collection

To evaluate FacialDataCollector (Method 2), we compare the program with other ways to handle data collection (Table 4). One other way (Method 1) is to manually simulate the data collection process, which requires a person to watch an entire video and list down the timings which the person deems to be a good data. Method 1 will be treated as one with 100% accuracy based on the assumption that the human can differentiate high-quality data from others. Method 3 will be one without the implementation of face error function but with pixel error function. Method 4 will be one without the implementation of pixel error function but with face error function.

S/N	Methods	$error_{face}$	$error_{pixel}$	Remarks
1	Manual data collection	N.A.	N.A.	Ground truth (most accurate method)
2	FacialDataCollector (FDC)	Yes	Yes	Current method
3	FDC w/o face error	Yes	No	No face error implemented
4	FDC w/o pixel error	No	Yes	No pixel error implemented

Table 4: List of methods for data collection

To compare, sample videos will be processed by each of the methods. The number of clips segmented with duration of more than four (4) seconds are counted, the standard for the duration of the clips, because they tend to elicit emotions more effectively (Uhrig, 2016). The

effectiveness of a method can be defined to be the percentage difference in the number of video clip collected as compared to Method 1. The lower the percentage difference, the more effective the method is, as it means frames which are not continuous are segmented from each other.

3.3 EVALUATION RESULT AND VALIDATION

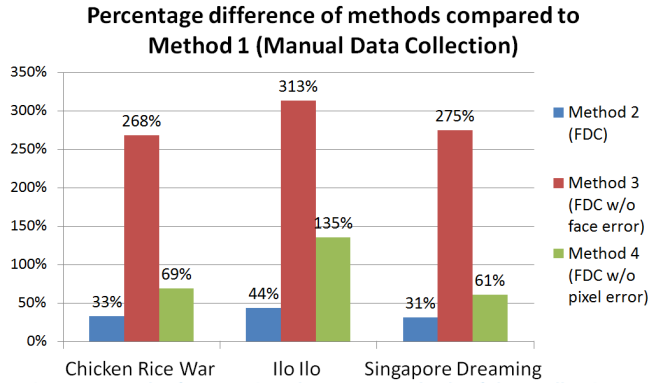


Figure 6: Graph of comparison between methods of data collection

It has been shown that FacialDataCollector (Method 2) consistently outperformed Method 3 and Method 4 by having a lower percentage difference (Figure 6). This shows the necessity and significance of combining both the error values from change in face positions and change in pixel values in the program.

Compared to Method 1, FacialDataCollector (Method 2) produces around on average 36.0% more videos than Method 1. This shows that FacialDataCollector is quite effective in reducing the number of fragmented videos and thus ensuring adjacent frames are continuous.

3.4 TECHNICAL ANALYSIS

The speed to which FacialDataCollector processes video clips is mainly limited by MoviePy, a movie processing library for Python, due to its inbuilt encoding and decoding of videos. The time for FacialDataCollector to segment video clips of less than 10 seconds is 10 seconds. However, the time to segment video clips of 30 seconds is around 20 minutes. On average, FacialDataCollector can analyse 120 minutes of video clips per day.

4 CONCLUSION

A data collector program FacialDataCollector was built. Such a tool can be helpful to researchers by helping them to extract subclips with frontal face elements from a video clip.

However, the downside of the program is that it is dependent on MoviePy, which severely limits its time-efficiency. To improve on the time-efficiency of FacialDataCollector, we suggest running FacialDataCollector on multiple computers so as to speed up the data collection process.

5 BIBLIOGRAPHY

- [1] Alvarado, N. (1997). Arousal and Valence in the Direct Scaling of Emotional Response to Film Clips. *Motivation and Emotion*, 21(4), 323-348. Retrieved November 21, 2017, from <https://www.cpp.edu/~nalvarado/PDFs/arousalValence.pdf>.
- [2] Fragkiadaki, K., Zhang, G., & Shi, J. (2012). Video Segmentation by Tracing Discontinuities in a Trajectory Embedding. Retrieved January 3, 2018, from https://www.cs.cmu.edu/~katef/papers/CVPR2012_discontinuities.pdf
- [3] Fragkiadaki, K., & Shi, J. (2011). Detection Free Tracking: Exploiting Motion and Topology for Segmenting and Tracking under Entanglement. Retrieved January 3, 2018, from https://www.cs.cmu.edu/~katef/papers/CVPR2011_topology_motion.pdf
- [4] Morales, M. R., Scherer, S., & Levitan, R. (2017). OpenMM: An Open-source Multimodal Feature Extraction Tool. *INTERSPEECH*, 3354-3358. <http://dx.doi.org/10.21437/Interspeech.2017-1382>
- [5] OpenCV. (2017, December 23). Retrieved January 1, 2018, from <https://opencv.org/>
- [6] Perner, P., & Rosenfeld, A. (2003). *Machine Learning and Data Mining in Pattern Recognition*.
- [7] Shapiro, T. (2016, October 17). How Emotion-Detection Technology Will Change Marketing. Retrieved December 22, 2017, from <https://blog.hubspot.com/marketing/emotion-detection-technology-marketing>
- [8] Tarantola, A. (2017, August 29). Robot caregivers are saving the elderly from lives of loneliness. Retrieved December 22, 2017, from <https://www.engadget.com/2017/08/29/robot-caregivers-are-saving-the-elderly-from-lives-of-loneliness/>
- [9] Uhrig, M. K., Trautmann, N., Baumgärtner, U., Treede, R., Henrich, F., Hiller, W., & Marschall, S. (2016). Emotion Elicitation: A Comparison of Pictures and Films. *Frontiers in Psychology*, 1-12. doi:10.3389/fpsyg.2016.00180

6 APPENDIX

6.1 PROGRAM IMPLEMENTATION

6.1.1 PROGRAMMING LANGUAGE

Python was selected as the language for FacialDataCollector because many of the preceding stages (YouTube parsers) and subsequent stages (OpenMM, AI) are accomplished using Python. This allows easier manipulation of the scripts, allowing smoother transitions between the stages.

6.1.2 GRAPHICAL USER INTERFACE (GUI) IMPLEMENTATION

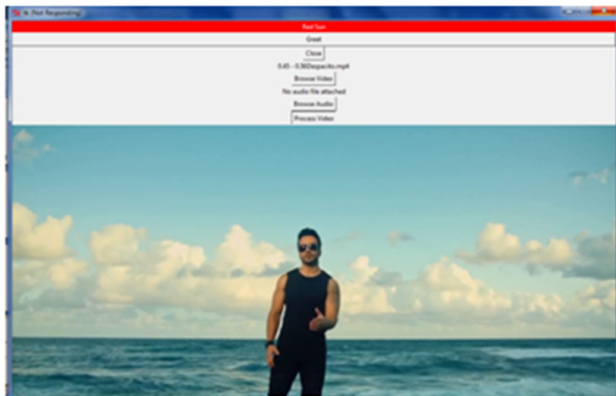


Figure 7: Graphical user interface processing clip 'Despacito'

A GUI is programmed to make the process of video segmentation more user-friendly. To select a video, a user just need to click "Browse Video". which allows him to preview the video (Figure 7). To segment the video, the user just need to click "Process Video". The video is then segmented and the resulting subclips are exported to an "export" folder.

6.1.3 USAGE OF FACIALDATACOLLECTOR

FacialDataCollector is maintained using a Git repository. To use the code, please fork the project FacialDataCollector from GitHub website: <https://github.com/jianzhi-1/FacialDataCollector>.